Guest-editorial

High Performance Java Compilation and Runtime Issues

Java-based technology has experienced an explosive growth in the computer sector since its inception in the early 1990s. Its meteoric rise has been fueled, in part, by solid language design choices that emphasize portability across diverse platforms – unfortunately, sometimes to the detriment of performance.

The High Performance Computing (HPC) community has recently considered Java as a vehicle for developing portable numeric codes. The HPC community is well-versed in strategies for achieving portability and high-performance; however, in general, achieving both simultaneously remains elusive or difficult at best goal. A popular technique for obtaining application portability is to isolate common number-crunching routines in a library and to define an architecture-independent library interface. Software and hardware techniques for obtaining high performance are often architecture or application specific and may be ad hoc. These aspects of high-performance Java codes are discussed in this issue.

The first four articles concern the use of an existing numeric library, LAPACK, by Java programs. In “The cost of being object oriented: A preliminary study” by Zoran Budimlić, Ken Kennedy and Jeff Piper, a straightforward translation of the Fortran version of LAPACK into Java is compared to one that exploits the object-oriented features of Java. An object-oriented Java implementation of LAPACK is also assessed in “An evaluation of Java for numerical computing” by Brian Blount and Sid Chatterjee. Tools for automatically generating Java interfaces to libraries are described and evaluated in “JLAPACK – compiling LAPACK Fortran to Java” by David Doolin, Jack Dongarra and Keith Seymour, and “Mulit-language programming environments for high performance Java computing” by Vladimir Getov, Paul Gray, Sava Mintchev and Vaidy Sunderam. The later work also applies their tool to other libraries, such as communication packages.

A Java interface for handling interprocessor communication is presented in “U-Net/SLE: A Java-based user-customizable virtual network interface” by Matt Welsh, David Oppenheimer and David Culler. The paper “Transient variable caching in Java’s stack-based intermediate representation” by Paul Týma describes a JIT compiler optimization, and the paper “Incorporating Intel MMX technology into a Java JIT compiler” by Aart Bik, Milind Girkar, and Mohammad Haghighat describes techniques for exploiting special-purpose hardware. “Java-based coupling for parallel predictive-adaptive domain decomposition” by Cécile Germain-Renaud and Vincent Néri reports on detailed optimizations for a specific application.

Susan Flynn-Hummel
IBM T.J. Watson Research Center
30 Saw Mill River Road, H1E20
Hawthorne, NY 10532
USA
Tel.: +1 914 784 7942
Fax: +1 212 242 7035
E-mail: hummel@watson.ibm.com